Dokumentácia k zápočtovému programu

Samuel Fanči

June 2021

Contents

1 Úvod

Toto je dokumentácia k "EVE market viewer", čo je konzolová aplikácia, ktorá dokáže interagovať s API MMO hry EVE online, a zobrazovať rôzne dáta.

Aplikácia sa skladá z jedného hlavičkového súboru `Interface.h`, ktorý obsahuje definície všetkých tried, a potom každá trieda ma vlastný `.cpp` súbor. Je tu taktiež `Source.cpp`, ktorý obsahuje funkciu `main()` a `run()`, ktoré viacmenej len vytvoria inštanciu hlavnej trieda a predajú jej input a output stream.

2 Triedy

2.1 MainInterface

Toto je trieda, ktorá obsahuje funkcie na spracovanie vstupu užívateľa, a taktiež inštancie tried ktoré interagujú s rôznymi časťami api

2.1.1 Konštruktor

Dostane output stream kde bude vypisovať informácie, a inicializuje `api_configuration`, čo je `http_config` z cpprest knižnice, na setup HTTP clienta.

2.1.2 `setup_api_client()`

Vytvorí vyššie spomínanú konfiguráciu

2.1.3 `setup_apis()`

Postupne vytvorí inštancie API tried pre podstatné rozhrania, pomocou `api_configuration`. Tieto APIs sú automaticky vygenerované z ESI(EVE API), a priamo s ním komunikujú, sú využívané ostatnými triedami.

2.1.4 `create_interfaces()`

Táto funkcia vytvorí inštancie rozhraní, ktoré sa používajú vo zvyšku programu. Každé z nich má ako argument okrem iného príšlušné API, a taktiež odkaz na `MainInterface`.

2.1.5 `parse_command`

Spracuváva stringy zo vstupu podľa prvého slova, a potom odkazuje na ostatné parsovacie funkcie.

2.1.6 Parsovacie funkcie

Ostatné parsovacie funkcie fungujú všetky podobne, najprv pomocou regexu zistia počet argumentov, a každá v `try` bloku skúsi zistiť ID príslušných názvov zo vstupu, a zavolá príslušnú funkciu na vypísanie výsledku. Ak je neplatný vstup, tak o tom informuje užívateľa.

2.1.7 `list_order_parser()`

Viď. Parsovacie funkcie

2.1.8 `history_parser()`

Viď. Parsovacie funkcie

2.1.9 `set_parser()`

Táto funkcia nastaví default na parametre zadané na vstupe. Tieto defaults sa používajú ak užívateľ nešpecifikuje nepovinné parametre.

2.1.10 `authorize_parser()`

2.1.11 `is_id()`

Pomocná boolovská funkcia, ktorá vráti či je daný field ID, alebo nie, podľa vstupného zoznamu `id_fields`.

2.2 MarketInterface

Táto trieda má na zodpobednosť interakciu s Market časťou ESI, čo znamená že dokáže stiahnuť informácie o cenách rôznych predmetov v hre, historické dáta a iné. Používa triedy `Orders` a `History` na enkapsuláciu a narábanie s týmito dátami v iných častiach aplikácie.

2.2.1 Konštruktor

Len nastaví odkaz na `MainInterface` a nejaký `ostream`,

2.2.2 `set_station()`

Nastaví default stanicu, v ktorej zobrazovať dáta

2.2.3 `set_system()`

Nastaví default soláru sústavu, v ktorej zobrazovať dáta

2.2.4 `set_region()`

Nastaví default región, v ktorom zobrazovať dáta.

2.2.5 `parse_orders()`

Načíta všetky buy/sell orders daného predmetu v danom regióne, a uloží ich v inštancií `Orders`. Keďže ESI vracia informácie po "stranách" v tomto prípade, prejde všetky strany kým to ide.

2.2.6 `get_type_orders()`

Funkcia, ktorá sa stará o prístup k orders daného typu, podľa toho ako špecificky to užívateľ chce. Môže to byť orders len v danej stanici, systéme alebo regióne.

Naneštastie ESI sa vie zamerať len na nejaký región, a nijak nezoraďuje dáta, takže je potrebné zistíť región v ktorom sa daná stanica/systém nachádza, a potom odfiltrovať všetky orders ktoré do toho nespadajú. Na toto sa využívajú funkcie `UniverseInterface`, a potom sa vráti pointer na objekt `Orders`.

2.2.7 `get_type_history()`

Načíta históriu ceny a objemu daného predmetu v danom regióne. Tu netreba filtrovať, pretože to veľmi nemá zmysel, keďže nie je ťažké chodiť medzi stanicami/systémami v danom regióne, a každý väčšinou obsahuje len malý počet hlavných "trading hubs", kde sú ceny veľmi porovnateľné. Vracia pointer na objekt `History`.

2.3 UniverseInterface

Trieda slúžiaca hlavne na hľadanie ID z názvov mien a naopak. Obsahuje cache, kde si ukladá výsledky predošlých requestov.

2.3.1 Konštruktor

Len priradi referenciu na `UniverseApi` od `MainInterface`

2.4 `get_name_from_id()`

Táto funkcia má za úlohu nájsť ID zodpovedajúcemu stringu. Ak je to `inventory_type`, tak to bude brať len exaktné (bez diakritiky) zhody, inak nájde prvý string, ktorý obsahuje dané meno ako substring, a vráti ID tejto veci.

2.4.1 `get_name_from_id()`

Podobne ako predošlá, len trochu jednoduchšia kedže poznáme ID.

2.4.2 `get_type_info()`

Nájde informácie o danom type, a vráti ich ako JSON.

2.5 SearchInterface

Malá trieda ktorá narába so Search sekciou ESI. Používaná na nájdenie najbližšej zhody.

2.5.1 `search()`

Pre zadaný `query` daného typu nájde zoznam ID typov, ktorých názov obsahuje `query` ako substring.

2.6 Orders

Orders je trieda na jednoduchsie manipulovanie so zoznamom market orders pre nejaký typ.

2.6.1 `sort()`

Sort pochopiteľne zoradí buy a sell orders podľa ceny zostupne pre buy orders, vzostupne pre sell orders.

2.6.2 `compare_numeric_field_ascending()`

Táto funkcia robí presne to čo je jej názvom, vytiahne číslenú hodnotu z price kategórie, a porovná tieto dva orders vzostupne.

2.6.3 `compare_numeric_field_descending()`

Rovnako.

2.6.4 `print()`

Táto funkcia má nastarosti pekné vytlačenie obsahu do konzole. Dá sa nastaviť šírka tabuľky ako parameter. Využíva na to `Printer`

2.7 History

Podobne ako `Orders`, táto trieda má nastarosti manipuláciu s historickými dátami.

2.7.1 `print()`

Rovnako ako pri `Orders`. Využíva na to tiež `Printer`, a dá sa nastaviť šírka tabuľky.

2.8 Printer

Hlavná trieda na vypisovanie JSONov do konzole.

2.8.1 json_field_to_string()

Vráti obsah nejakého fieldu ako `string`, aj keď je to číslo. Pomocná funkcia.

2.8.2 `print_head()`

Vypíše hlavičku tabuľky, teda názvy kategórií ktoré sa budú vypisovať nižšie.

2.8.3 `print_line()`

Vypíše čiaru tak dlhú, aby to bolo zarovnané s veľkosťou tabuľky, alebo sa dá dať špecifická šírka.

2.8.4 `print_help`

Vypíše obsah `help.txt`, čo obsahuje help pre tento program.

2.8.5 `print_description`

Pekne vypíše popisok daného typu, zarovnaný na 80 znakov, spolu s menom.

2.8.6 `print_description`

Na vstupe dostane JSON vector, ktorý vypíše ako tabuľku. Treba zadať fields ktoré sa majú vypísať ako argument, taktiež ako ID fields, pretože tie treba premeniť na ich názvy. Dá sa nastaviť šírka a počet riadkov.